Knapik Kinga 171046
Stoczek Weronika 171054
Stupiec Emil 163770

Seminarium dyplomowe 2011

Mechanizmy systemu operacyjnego wspomagające synchronizację procesów

System operacyjny stanowi oprogramowanie zarządzające sprzętem komputerowym, tworzące środowisko do uruchamiania i kontroli zadań użytkownika. Zatem jedną z podstawowych jego funkcji jest zarządzanie procesami.
Program to obiekt pasywny, zbiór instrukcji dla procesora przechowywany na dysku w postaci pliku.
Proces to wykonujący się program, jest obiektem aktywnym i podstawową jednostką pracy systemu operacyjnego. Każdy proces, aby w pełni wykonać swoje zadanie wymaga przydziału pewnych zasobów, włączając w to czas procesora, pamięć, pliki oraz dostęp do urządzenia wejścia/wyjścia. Obsługą procesów zajmuje się jądro systemu operacyjnego. Z punktu widzenia procesora zawsze wykonywany jest jeden program. Jednak jądro systemu operacyjnego przeplatając instrukcje procesów stwarza wrażenie, że wiele procesów wykonuje się jednocześnie.
Problem synchronizacji procesów pojawia się wszędzie tam, gdzie mamy do czynienia ze współpracującymi ze sobą współbieżnymi procesami. Najczęstszymi przypadkami, w których konieczna jest synchronizacja procesów są:

a) Procesy współdzielą pewną strukturę danych – pojedyncze operacje muszą być zatomizowane, tzn. tylko jeden proces może na raz modyfikować współdzieloną strukturę danych. Jest to przykład problemu sekcji krytycznej.
b) Wyniki działania jednego procesu stanowią dane dla innego procesu – drugi z procesów może przetwarzać dane dopiero wtedy, gdy zostaną one obliczone przez pierwszy z procesów. Taką zależność między procesami nazywamy problemem producenta-konsumenta.
c) Procesy korzystają z pewnej wspólnej puli zasobów, które pobierają i zwalniają wedle potrzeb. Przykładem ilustrującym taki rodzaj synchronizacji jest problem pięciu filozofów.

Sekcja krytyczna to ciąg operacji wykonywanych na pewnym zasobie (zwykle pamięci), który musi być wykonywany w trybie wyłącznym przez tylko jeden z potencjalnie wielu procesów. Warunki poprawnego rozwiązania sekcji krytycznej:

a) W sekcji krytycznej może być tylko jeden proces, to znaczy instrukcje z sekcji krytycznej nie mogą być przeplatane.
b) Nie można czynić żadnych założeń, co do względnych szybkości wykonywania procesów.
c) Proces nie może się zatrzymać w sekcji krytycznej.
d) Zatrzymanie procesu w sekcji lokalnej nie może blokować innym procesom wejścia do sekcji krytycznej.
e) Każdy proces musi w końcu wejść do sekcji krytycznej.

Mechanizmy systemu operacyjnego wspomagające synchronizację procesów:
Zmienne współdzielone to zmienne używane do synchronizacji procesów współbieżnych, do których wszystkie procesy mają dostęp. Służą one do kontroli dostępu do sekcji krytycznej. Większość rozwiązań pozwala skutecznie rozwiązać problem synchronizacji tylko dla ustalonej

wcześniej liczby procesów współbieżnych i opiera się na aktywnym czekaniu, więc jest mało efektywne.
Algorytm wzajemnego wykluczania (mutex) stanowi szczególny rodzaj semaforów binarnych. Muteks może być zablokowany (ma wartość 1) lub odblokowany (ma wartość 0 - niezablokowany). Muteksy w odróżnieniu od semaforów posiadają pewne mechanizmy zabezpieczające. Mianowicie jeśli jakieś zadanie zablokuje muteks (nada mu wartość 1), to tylko ono może ten muteks odblokować (nadać mu wartość 0). Takie podejście eliminuje całkowicie problem niespójności danych, w którym jedno z zadań wykona operację czekaj, aby synchronizować dostęp do jakiegoś zasobu a później inne, niezwiązane logicznie z tym zasobem, podniesie semafor. Może to prowadzić do uszkodzenia wspólnych danych. Co więcej muteksy posiadają możliwość blokowania w sposób rekurencyjny polegający na określeniu ile razy zadanie, które posiada dany muteks wykonało na nim operację blokowania. Jest to bardzo przydatna cecha, która umożliwia uniknięcia takich sytuacji jak np. zakleszczenie procesów .
Semafor Dijkstry jest obiektem abstrakcyjnym służącym do kontrolowania dostępu do ograniczonego zasobu. Z semaforem związany jest licznik zasobu przyjmujący wartości nieujemne, a w przypadku semafora binarnego wartości 0 lub 1. Wyróżniamy dwa rodzaje semaforów:

a) Semafor ze zbiorem procesów oczekujących – nie jest określone, który z oczekujących procesów ma być wznowiony. Nie spełnia on warunków zagłodzenia, tzn. któryś z procesów może nigdy nie wejść do sekcji krytycznej.
b) Semafor z kolejką procesów oczekujących – procesy oczekujące na semaforze umieszczane są w kolejce FIFO.

Przykład zastosowania semafora do ochrony sekcji krytycznej:

```
semaphore S; //Deklaracja semafora
sem_init(S,1); //Inicjalizacja semafora
Proces1 {                    Proces2 {
  do {                         do {
    Sekcja_lokalna;              Sekcja_lokalna;
    sem_wait(S);                 sem_wait(S);
    Sekcja_krytyczna;            Sekcja_krytyczna;
    sem_post(S);                 sem_post(S);
  } while(1);                  } while(1);
}                            }
```

Nie da się z góry określić czasu działania całej aplikacji współbieżnej zawierającej sekcję krytyczną zabezpieczoną semaforem. Gdy semafor jest zajęty inne procesy współbieżne muszą poczekać na jego zwolnienie i nie mogą kontynuować swojej pracy. Standard POSIX jednak umożliwia podjęcie nieblokującej próby zajęcia semafora. W przypadku gdy semafor był już zajęty, proces kontynuuje swoją pracę bez korzystania ze współdzielonego zasobu.
Semafory mogą prowadzić do zakleszczeń, w których jeden proces dostaje semafor, kiedy inny proces dostaje drugi semafor, może dojść do zakleszczenia gdzie oba procesy czekają na wzajemnie przez siebie zajęte semafory. Inny efekt uboczny polega na zagłodzeniu, w którym proces nigdy nie otrzyma wystarczającej ilości zasobów do wykonania polecenia. Może się także zdarzyć inwersja priorytetów, w której wątek o wyższym priorytecie czeka na wątek z niższym priorytetem.
Kolejki komunikatów - komunikaty uporządkowane są w kolejce FIFO. Dodany komunikat umieszczany jest zawsze na końcu kolejki natomiast wyjmowany jest z kolejki zawsze element umieszczony w niej najdawniej. Na kolejce wykonywać można podstawowe operacje takie jak wkładanie i wyciąganie zmiennych. Pojemność kolejki jest ograniczona. Występują różne

implementacje tego mechanizmu. Przykładowo kolejki w których procesy piszący jest zawieszany jeżeli kolejka jest przepełniona lub implementacja, w której zapis po prostu nie dokona się i pojawi się odpowiedni komunikat o błędzie. Istnieje możliwość zaimplementowana kolejki komunikatów z wykorzystaniem semaforów oraz odwrotnie tzn. implementowania semaforów poprzez wykorzystanie kolejki

Monitor jest strukturalnym narzędziem synchronizacji. Są to zmienne i działające na nich procedury zebrane w jednym module. Monitory definiowane są jako klasy. Dostęp do zmiennych monitora jest możliwy tylko za pomocą procedur monitora. W danej chwili tylko jeden proces może wykonywać procedury monitora. Gdy inny proces wywoła procedurę monitora proces ten będzie zablokowany do chwili opuszczenia monitora przez znajdujący się tam już proces. Istnieje możliwość wstrzymania i wznowienia procedur monitora za pomocą zmiennych warunkowych. Procesy oczekujące na wejście do monitora są zorganizowane w kolejkę FIFO. Na zmiennych warunkowych można wykonywać operacje:

a) wait – powoduje wstrzymanie procesu i wstawienie go na koniec kolejki.
b) signal – powoduje wznowienie pierwszego procesu z kolejki, o ile kolejka nie jest pusta, jeżeli jest pusta, operacja nie przynosi żadnego efektu.